Desarrollo FrontEnd con JavaScript

s05 – Backbone.js : Modelos y colecciones

Ubicuidad

Facilidad para actualizar

Aplicaciones Web

Uso intensivo de JavaScript

Uso de patrones y bibliotecas

REST

REST

- Arquitectura simple sobre HTTP.
- Uso de recursos como unidad principal para operaciones.
- Operaciones CRUD => Operaciones HTTP.

```
Create : POST
Read   : GET
Update : PUT
Delete : DELETE
```

RPC vs REST

```
createUser()
getUser()
updateUser()
deleteUser()
```

```
POST   /users
GET    /users/1
PUT    /users/1
DELETE /users/1
```

Backbone.js

http://backbonejs.org

Biblioteca, no framework

Creación de aplicaciones web

Backbone.js

Operaciones CRUD + REST

MV*

Backbone.js : Modelos

Backbone.Model

```
var Book = Backbone.Model.extend({
  urlRoot: '/books',
  getPriceIn: function(currency) {
    var newPrice = this.get('price') || 0.0;

    if (currency == 'PEN') {
      newPrice = newPrice * 2.79;
    }

    return newPrice;
  }
});
```

Backbone.Model

- Representa una unidad básica de la aplicación y maneja su lógica correspondiente.

```
var book = new Book({
  price: 27.99
});

book.getPriceIn('PEN');
```

Backbone.Model

- Ofrece soporte para eventos.
- Sincronización configurable.

```
book.save({
  success: function(model, response, options) {
    alert('The book has been saved');
  },
  error: function(model, xhr, options) {
    alert('Error: ' + xhr.statusText);
  }
});
```

Backbone.js : Colecciones

Backbone.Collection

```
var BookCollection =
Backbone.Collection.extend({
  url: '/books',
  model: Book
});

var myLibrary = new BookCollection();
myLibrary.add(book);
```

Backbone.Collection

- Contiene métodos para trabajar con listas de modelos (`sort`, `map`, `filter`).

```
var book = myLibrary.get(123);
var ids = myLibrary.pluck('id');

var sortedByPrice =
myLibrary.sortBy(function(book) {
  return book.get('price');
});

var topBooks =
myLibrary.filter(function(book) {
  return book.get('rating') >= 4.5;
});
```

Backbone.Collection

- Ofrece soporte para eventos: change, add y remove.
- Sincronización configurable.

```
BookCollection.prototype.sync =
function() {
  this.each(function(book) {
    var key = 'Book:' + book.get('id');
    var value =
JSON.stringify(book.toJSON());
    window.localStorage.setItem(key,
value);
  });
};
```